Linear Systems continued...

(Beyond Gaussian elimination).

In the last lecture we developed the LU decomposition based on Gaussian elimination (also with "scaled partial pivoting."). The upper triangular matrix U corresponded to the coefficients of the linear system after the Gaussian elimination steps, while the lower triangular matrix L contained all of the "multipliers" m_{ij} used during the Gaussian elimination process (plus 1's along the diagonal).

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & & & \\ m_{21} & 1 & & \\ m_{31} & m_{32} & \ddots & \\ \vdots & \vdots & & \\ m_{n1} & m_{3n} & \cdots & m_{n,n-1} \; 1 \end{pmatrix}}_{L} \underbrace{\begin{pmatrix} \hat{a}_{11} & \hat{a}_{12} & \cdots & \hat{a}_{1n} \\ & \hat{a}_{22} & & \\ & & \ddots & \\ & & & \hat{a}_{nn} \end{pmatrix}}_{U}$$

Note that we can determine the matrices L and U even before we know the right hand side of the equation (i.e. b):

$$Ax = b$$

If we are now given <u>multiple</u> linear systems to solve which all exhibit the same left hand side (same matrix A) but different right hand sides (differing vectors b), we can exploit the LU decomposition to solve each of these systems efficiently by writing them in the form

$$L\underbrace{(Ux)}_{z} = b$$

forward substitution

step 1: Solve $Lz = b$ for z

step 2: Solve $Ux = z$ for x

backward substitution

EX An example of solving multiple systems with the same A but differing b's is found in computing the inverse A^{-1} of a matrix.

$$A \underbrace{A^{-1}}_{\text{unknown}} = I$$

$$A \begin{pmatrix} x_{11} \\ x_{21} \\ \vdots \\ x_{n1} \end{pmatrix} \begin{pmatrix} x_{12} \\ x_{22} \\ \vdots \\ x_{n2} \end{pmatrix} \cdots \begin{pmatrix} x_{1n} \\ x_{2n} \\ \vdots \\ x_{nn} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix} \cdots \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}$$

$\vec{x}_1 \quad \vec{x}_2 \cdots \vec{x}_n \qquad I_1 \quad I_2 \cdots I_n$

$$\underbrace{A\vec{x}_1 = I_1, \quad A\vec{x}_2 = I_2, \quad \cdots, \quad A\vec{x}_n = I_n}_{n \text{ systems to solve}}$$

Step 1: First compute $A = LU$

Step 2: Now solve the more efficient systems

$$Lz_i = I_i \quad \text{(solve for } z_i\text{)}$$
$$Ux_i = z_i \quad \text{(solve for } x_i\text{)}$$

for each $i = 1, 2, \cdots, n$

LDU decomposition:

For even further symmetry, we can divide each row of the matrix U (from the Gaussian elimination process) by its diagonal element d_{ii} (for row i) so that the new matrix U has 1's along the diagonal just like the matrix L. If we then create a purely diagonal matrix D to store the original diagonal elements d_{ii} of the original U-matrix, we obtain the well-known LDU decomposition

$$A = L \cdot D \cdot U$$

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & & \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} 1 & & & \\ l_{21} & 1 & & \\ l_{31} & l_{32} & 1 & \\ \vdots & & & \\ l_{n1} & l_{n2} & \cdots & l_{n,n-1} \; 1 \end{pmatrix} \begin{pmatrix} d_{11} & & & \\ & d_{12} & & \\ & & \ddots & \\ & & & d_{nn} \end{pmatrix} \begin{pmatrix} 1 & u_{12} & u_{13} & \cdots \\ & 1 & u_{23} & \cdots \\ & & 1 & \cdots \\ & & & \ddots \end{pmatrix}$$

Cholesky Factorization

Theorem: If an $n \times n$ matrix A is symmetric and positive definite

there exists a unique factorization $A = LL^T$ where L is lower triangular with positive diagonal elements.

NOT the same L

Note that we may relate the LDU factorization to the Cholesky factorization as follows

$$A = LDU = L D^{1/2} D^{1/2} U = L D^{1/2} D^{1/2} L^T$$

Only happens when we have A symmetric positive definite

$$= \underbrace{(L D^{1/2})(L D^{1/2})^T}$$

Cholesky

↳ only need to store $\frac{n(n+1)}{2}$ elements rather than n^2

QR Factorization:

Another decomposition useful for solving least squares problems in particular is to zero-out the subdiagonal elements of A by multiplication on the left by an orthogonal matrix Q ($Q^{-1} = Q^T$).

Although the resulting matrix $Q^T A$ is upper triangular, it will not necessarily be the same as that obtained by Gaussian elimination (LU). We therefore use the symbol R instead. Furthermore the matrix Q is usually NOT in lower triangular form:

$$A = \underbrace{Q}_{\text{orthogonal}} \underbrace{R}_{\text{upper triangular}}$$

$$\downarrow$$

$$QQ^T = I$$

This decomposition always exists but is Not unique!

Iterative Improvement

Suppose we have solved the system $Ax = b$ using one of the previous "direct methods" (LU, LDU, LL^T, or QR). Except for small matrices (small n), there are likely to be some numerical errors as a result of the cummulative effect of finite precision arithmetic.

We can at least partially correct for this by noting that our solution has the form

$$x + \delta x$$

(x: true solution; δx: numerical error)

and that

$$A(x + \delta x) = b + \delta b$$

subtracting $Ax = b$ yields

$$\boxed{A\,\delta x = \delta b}$$

Solve for δx by noting

$$\delta b = A(\underbrace{x + \delta x}_{\text{first solution}}) - b$$

Once we solve for δx, we then subtract it from our original answer to obtain an improved estimate of the solution X. This process can be repeated if further refinement is necessary.

Fully Iterative Methods

For very large, sparse systems (tri-diagonal for example), iterative methods are often employed to solve $Ax = b$ to reduce the storage requirements and/or the numerical inaccuracies that accumulate when applying "direct methods" to large systems. The simplest method to consider is "gradient descent"

Note that the solution of $Ax = b$ is also the minimizer of

$$f(x) = \frac{1}{2} x \cdot (Ax) - b \cdot x$$

whose gradient ∇f is

$$\nabla f = Ax - b$$

(note the gradient is zero when $Ax = b$).

If we start with an initial guess X_0, then we determine the gradient $\nabla f(x_0)$ and use this as a search direction

$$p_0 = Ax_0 - b$$

(x_0: current estimate; p_0: search direction (starting from x_0))

Now we minimize $f(x)$ along the direction p_0 through the point x_0 to obtain the new estimate x_1:

$$x_1 = x_0 + \underbrace{\arg\min_{\alpha} (f(x_0 + \alpha p_0))}\, p_0$$

$\alpha_0 = ?$ ← to find α_0 we compute

$$\frac{\partial}{\partial \alpha} f(x_0 + \alpha p_0)\Big|_{\alpha_0} = 0$$

$$0 = \frac{\partial}{\partial \alpha} f(x_0 + \alpha p_0)\Big|_{\alpha_0} = \frac{\partial}{\partial \alpha}\left[\frac{1}{2}(x+\alpha p_0)^T A(x_0+\alpha p_0) - b^T(x+\alpha p_0)\right]\Big|_{\alpha_0}$$

$$= (x_0 + \alpha_0 p_0)^T A^T p_0 - b^T p_0$$

and so

$$\alpha_0 = \frac{(b^T - x_0^T A^T)\, p_0}{p_0^T A^T p_0} = \frac{(b - Ax_0)\cdot p_0}{p_0 \cdot Ap_0}$$

giving us:

$$x_1 = x_0 + \frac{(b - Ax_0)\cdot p_0}{p_0 \cdot Ap_0}\, p_0$$

$$x_1 = x_0 + \frac{(Ax_0 - b)\cdot(Ax_0 - b)}{(Ax_0 - b)\cdot A(Ax_0 - b)}\,(Ax_0 - b)$$

we can now iterate this relationship by replacing the symbol x_0 with x_k and x_1 with x_{k+1}

Unfortunately this method is very slow to converge. In the case that A is symmetric and positive definite, we may modify the search direction p_k so that it is perpendicular to all previous search directions and such that the new estimate x_{k+1} will be minimized not only along the new direction p_k but also over all previous directions $p_{k-1}, p_{k-2}, \ldots, p_0$ (this "magic" comes from the symmetric + positive definite structure of A). The resulting algorithm is called conjugate gradient descent and has the following iterative structure.

Conjugate Gradient Method

$$x_{k+1} = x_k + \alpha_k p_k$$

where $r_{k+1} = r_k - \alpha_k A p_k$ $\left(= b - Ax_{k+1}\right)$ ← "new residual"

(just substitute in order to prove this.)

$$\alpha_k = \frac{r_k \cdot r_k}{p_k \cdot A p_k}, \quad \beta_k = \frac{r_{k+1} \cdot r_{k+1}}{r_k \cdot r_k}$$

$$p_{k+1} = r_{k+1} + \beta_k p_k \leftarrow \text{"new search direction"}$$

If the matrix A is $n \times n$, then this method theoretically gives the exact solution after at most n iterations (due to numerical errors, however, the solution might not always be "exact" by instead "very close").